Jacob Hackett
CS4341 Intro to AI
Project 3 Writeup

Feature Generation

1. The first feature depended on which player had control of the bottom left corner of the board. This was, by far, the easiest to calculate and I just had to return that zeroth index of the board. I do not think that this is a good feature because owning the bottom left corner in connect four is perhaps one of the least important places to win. There are only three ways to win from there and they are all very easily and often blocked by the opponent. This was a required feature. The player with control of the bottom left spot won.
2. The second feature was also one of the required features. It depended on which player owned more pieces in the middle rows of the board. This is a better heuristic than the previous one but still isn't ideal because it does not weigh pieces closer to the center higher than those farther away. In order to calculate this feature, I iterated over the board and kept track of the number of pieces each player had in these rows. The player with the greater number of pieces in the middle 2 rows won.
3. The third feature was similar to the second feature but involved the center rows instead of columns. This has a greater emphasis on the center of the board but also does not take into account the top few rows being less important than the bottom ones. In order to calculate this feature, I iterated over the middle of the board and kept track of the number of pieces each player had in these columns. The player with the most number of pieces in the middle 3 columns won.
4. The fourth feature depended on which player had greater board control. I defined board control by the number of pieces the player has at the top of each column. I believe a player is more likely to win if they have greater board control and am interested to see how well this feature works. In order to calculate this feature, I iterated over the board and kept track of the number of pieces each player had at the top of each column. The player with the most pieces at the top of the columns won.
5. The fifth feature depended on which player had more pieces closer to the center. It calculated the manhattan distance for each piece to the center of the board. For this I used the piece in index 20 as the center of the board. So this actually slightly favors the bottom center of the board. I think this will be the strongest performer because it takes into consideration the things that made the first three features weak. In order to calculate this feature, I iterated over the board and kept track of the distance that each piece was from the center piece. The player with the greater number of closer pieces won.

Decision Tree

In order to test the decision tree, I was able to run it many different times with different holdout percentages. I observed the number of correct classifications against the total number of data points. I found that feature 1 was virtually useless. It rarely ever came up in a combination or alone as a successful feature. However, I predicted this in the descriptions above. I was extremely surprised to see that feature 5, the manhattan distance feature, was also very rare. Almost all of my simulations results in a combination of features 2, 3, and 4. I was also fairly surprised that even with the combinations of features, the program did not perform very well. It would predict the correct winner somewhere between 35% and 75% of the time.

One of the important things here is that I use cross validation. It is extremely easy to overfit the data, or just memorize the exact states that we have. That means that when you go to predict other states that you haven't seen before, it make worse predictions. You use cross validation to combat this by having a training data set and a testing data set. You train with a percent of the input and then test your training against the testing set. You can continue doing this many times until results are satisfactory.

	Feature 1		Feature 2		Feature 3		Feature 4		Feature 5	
	No Cross Validation	Cross Validation	No Cross Validation	Cross Validation	No Cross Validation	Cross Validation	No Cross Validation	Cross Validation	No Cross Validation	Cross Validation
# correct	233	186	351	280	493	394	379	299	225	181
# incorrect	767	614	649	520	507	406	621	501	775	619
% error	76.7	76.75	64.9	65	50.7	50.75	62.1	62.625	77.5	77.375

We only see small differences in the percent error here. This could be because we have very few features or that the features are not good enough. But it seems like the program is not overfitting the data very much.

In order to find what features were most useful, I ran this program several times with several different holdout percentages, as I did above for this data. I looked at the generated feature paths and made conclusions based on what features were included and how often. Features 1 and 5 were almost never included and features 2, 3 and 4 were always included.

Improved Connect-4 Heuristic
After looking at the results of the decision tree experiments. I determined that it would be wise to include at least one of features 2, 3, or 4 in my heuristic for connect-4. Previously it just looked at the number of unblocked connections on the board. Now it also takes into consideration the number of pieces each side has in the middle columns. I decided to use this feature because it appeared often in my tests and was relatively easy to implement.